REMARKS/ARGUMENTS

Claims 5 and 17 have been amended. Support for these amendments can be found in paragraphs 46 and 47 of the specification. Claims 1-36 remain pending for examination.

Rejections Under 35 U.S.C. § 103(a)

Rejection Of Claims 1-12 And 25-36

The Office Action has rejected claims 1-12 and 25-36 under 35 U.S.C. § 103(a) as being unpatentable over the cited portions of U.S. Patent Publication No. 2002/0129129 to Bloch et al. ("Bloch") in view of a publication entitled "SDML - Signed Document Markup Language" ("SDML"). Claims 13-24 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Bloch in view of U.S. Patent No. 6,314,451 to Landsman ("Landsman"). The Applicant believes that a *prima facie* showing of obviousness has not been properly set forth as claim limitations are not expressly or implicitly taught in the references.

The cited prior art does not explicitly teach or suggest the recitation of "checking automatically for updated versions of said text files" as required by claims 1-12 or "receiving automatically updated versions of said text files" as required by claims 25-36. The Office Action confirms that this recitation is not explicitly taught by the prior art. Office Action, page 2. The Applicant respectfully points out that Bloch does not explicitly disclose the recitation of checking automatically for updated versions of text files.

Bloch discloses a system for deploying applications over a distributed network to web-enabled devices. Bloch, ¶ 12. This system includes a server and an assembler program. Id. The assembler program is referred to throughout the disclosure as an application virtual machine (AVM). The assembler program can be downloaded and installed on a web-enabled device. Id. The server includes stored text files that are embedded with application logic that may be downloaded from the server by the AVM, assembled and organized into a functioning application. Id.

Bloch discusses upgrading and updating the AVM, but does not discuss automatically "updating" and/or "upgrading" the text files as claimed. The cited art describes "updating" in paragraphs 53, 63, 82 and 100 and "upgrading" in paragraphs 32, 53, 61 and 100. Paragraph 53 describes updating the AVM, but does not describe updating the text files on which the AVM operates. Paragraph 63 describes allowing a user to update preferences, which is different than checking for updates of text files. Paragraph 82 discloses updating a document tree with downloaded data, such as a downloaded text file, but does not disclose automatically checking for updates to these text files. Paragraph 100 describes how the AVM updates data based on input from a user rather than checking for updates to text files. Nowhere does Bloch describe automatically checking for updates of text files.

Turning to "upgrading," which in some contexts may be implicated by "updating," paragraph 32 allows for upgrades of various components of the AVM, but does not discuss automatically checking for updates to text files. Paragraphs 53 and 107 each disclose upgrading the AVM rather than the text files. These paragraphs, therefore, cannot and do not implicate checking for updates to the text files.

Paragraph 61 of Bloch does discuss upgrading text files, but does not discuss doing so automatically. Instead, Bloch "downloads the XML Files 144 and Image Files 145 and assembles the component elements each time the user visits the site, so that additions and modifications can be introduced on the fly." Rather than "checking automatically for updated versions of said text files," Bloch downloads and assembles the files each time a user visits a specific website. Thus, Bloch does not need to automatically check for updated versions of text files because these files are downloaded each time the user visits a website whether or not the text files have been updated. This functionality eliminates the need to automatically check for updates.

The Office Action seems to suggest that automatically checking for updated versions of text files is implicit in the teaching of Bloch as further shown by SDML. The Office Action asserts that Bloch "addresses the urge for providing latest set of files in accordance to appropriate version of script file or virtual machine files with regard to version." Office Action, pages 2-3. Bloch does include a scheme for providing the latest versions of files as described in paragraph 61. However, the scheme used by Bloch to ensure that the latest versions of files are downloaded is different than what is disclosed by the claims. Specifically, Bloch "downloads the XML Files 144 and Image Files 145 and assembles the component elements each time the user visits the site, so that additions and modifications can be introduced on the fly." Downloading files each time a user visits a site removes any need to check for updates for such files, because updated versions of these files have just been downloaded. It would be redundant for Bloch to automatically check for an updated version of the file that was just downloaded.

The Office Action further points out that version information included in the header of XML, HTTP or any markup language file is further evidence of the implicit nature of automatically checking for updated versions of text files. Office Action, page 3. A limitation can only be implied as a teaching from a reference if that limitation necessarily follows from what is expressed in a reference. The fact that version information may be included in headers of markup language files does not necessarily require automatic checking of updated versions of files. Version information may be included in markup language files for reasons other than automatically checking for updates. Some applications may only operate properly on XML files that include specific versions. Moreover, version information may be included in the headers of markup language files to aide in compliance checking, debugging, or customer service concerns. Version information may be included in headers to aide in checking for updated versions of files, but it is not necessarily so.

Furthermore, including version information in a header is common programming practice. Programmers typically provide version information in programs and program files like XML and HTML. As such, the fact that version information is included in a header of an XML file cannot necessarily imply the need to automatically check for updates of such files.

The Office Action further implies that compliance checking of markup language files by a browser engine, as exemplified by SDML, would make automatic checking for updated text files obvious to one skilled in the art. Office Action, page 3. All SDML shows is that markup language documents include version information. There is nothing in SDML that suggests that this version information necessarily requires a system to automatically check for updates. The version information may be used for this purpose, and may be used by a browser engine to do so, but automatically checking for updated versions does not necessarily follow from XML documents with version information.

With regard to claim 5, Bloch does not disclose executing the application in asynchronous mode. Operating in asynchronous mode includes "sending a request from said client computer system to said server computer system; and executing said application in parallel with and while waiting for a response to said request without interruption of application functionality." According to the claimed language, the application remains functional despite waiting for a request from a server computer system. Bloch does not teach an asynchronous mode as claimed. Bloch does discuss downloading data and periodically updating the Document tree, which is different from operating the application without interruption while waiting for data from a server. There is nothing in Bloch that teaches or suggests asynchronous operation as claimed.

Similarly, with regard to claim 6, Bloch is silent on executing the application in connectionless mode. Bloch does disclose an AVM that downloads stored text files embedded with application logic that may be downloaded from a server, assembled and organized into a functioning application. The AVM requires a connection in order to download the stored text files. The applications organized from the text files, on the hand, may or may not require a network connection in order to operate. There is nothing in Bloch that teaches or suggests that the application may operate in a connectionless mode.

There is ample disclosure in Bloch, however, that an application does require a network connection. For example, paragraph 38 discusses how the AVM accesses the server, as needed in response to interaction with the assembled application. Paragraph 39 describes how each interaction between a user and the application does not require communication with a server, however, the paragraph goes on to specifically point out that access to the server is required for some interactions. Requiring access for some interactions could not be performed in a connectionless environment as required by claim 6.

Rejection Of Claims 13-24

The Office Action also rejected claims 13-24 under 35 U.S.C. § 103(a) as being unpatentable over Bloch in view of the cited portions of U.S. Patent No. 6,314,451 to Landsman. The Applicant believes that a person of ordinary skill in the art would not have combined Bloch with Landsman as suggested by the Office Action. Moreover, no motivation is suggested by the Office Action.

Bloch is directed towards a system for deploying applications over a distributed network to web-enabled devices. Landsman, on the other hand, is directed towards an advertising management system that directs advertisements to user browsers. One of ordinary skill in the art looking to improve the system for deploying applications would not look towards references directing advertisements to consumers over the network. The commonality of network connectivity and directing either applications or advertisements is too tenuous in light of the differences between the two references. Downloading deployable applications running on XML as disclosed by Bloch is different than displaying HTTP coded advertisements in a web environment. Bloch downloads an AVM that may download and execute XML files while Landsman simply distributes HTTP coded advertisements. One of ordinary skill in the art would not seek to combine such references.

Moreover, claims 17 and 18 are similar to claims 5 and 6. As discussed above, Bloch does not teach the elements of operating in an asynchronous mode or operating in a connectionless mode. Accordingly, these claims are distinguishable on these additional grounds.

Conclusion

In view of the foregoing, the Applicant believes all claims now pending in this application are in condition for allowance and an action to that end is respectfully requested.

If the Examiner believes a telephone conference would expedite prosecution of this application, please telephone the undersigned at 303-571-4000.

Respectfully submitted,

Date: June 19, 2007

/Jason A. Sanders/
Jason A. Sanders
Reg. No. 59,984

TOWNSEND and TOWNSEND and CREW LLP
Two Embarcadero Center, Eighth Floor
San Francisco, CA 94111-3834
Tel: 303-571-4000
Fax: 415-576-0300
JAS/jln
61078237 v1